RESEARCH ARTICLE

Identity based Access Control for Pro-active Message's DTN

Sánchez-Carmona, Adrián*; Robles-Martínez, Sergi and Borrego-Iglesias, Carlos

Department of Information and Communications Engineering (dEIC), Autonomous University of Barcelona (UAB), 08193 Bellaterra, Spain

ABSTRACT

Pro-active message's DTN are based on the usage of mobile code to obtain messages that contain their own routing code. This architecture allows applications to use the same network in different ways. The keystone of this type of heterogeneous network is a collection of contextual and application-related information that it is stored in every node and accessed by the messages' routing code. Access to that information must be protected in order to make feasible the whole architecture, the operation of the network has to be secure, and attacks of information poisoning have to be avoided. We propose an Identity based Access Control system for Pro-active Message's DTN based on tools that are available in DTN networks, like symmetric key encryption and hashes. Our system grants confidentiality and integrity to the contextual information and solves the question of messages needing to use distributed information stored in nodes to route properly. The proof-of-concept of Identity based Access Control in a certain kind of application demonstrates the feasibility of the proposal. The comparison between our proposal and other access control systems shows that Identity based Access Control is the only system that fits well with the special characteristics of Pro-active message's DTN.

KEYWORDS

Cryptographic applications; Heterogeneous communications network security; Security for distributed networks; Cryptographic mechanisms; DTN Access Control; Security in DTN

***Correspondence**

adria.sanchez@deic.uab.cat

Received . . .

DOI: 10.1002/sec

Prepared using secauth.cls

Security Comm. Networks 2012; **00**:2–26

DOI: 10.1002/sec

1. INTRODUCTION

Delay Tolerant Networks (DTN) have a set of unique characteristics that make it very difficult to find a routing protocol that can be applied successfully in any situation. Some of them are the non-contemporary of communications, the existence of significant delays and mobility patterns that allow nodes to be isolated from their neighbours during variable lapses of time. Due to that, there is not a *de facto* routing standard broadly extended and used. This makes DTN routing a challenge.

In an environment where the topology of the network changes very quickly and the nature and characteristics of the applications that use it are very different, the one-fit-all solution could be a chimera. Flexibility is a key aspect, and the only proposal that tries to obtain it, Haggle [1], uses a dynamic approach with many pros and one important contra: the utilisation of an array of routings algorithms makes the deployment very hard and costly in any scenario where the set of applications grows, or simply changes, over time.

In response to that, we propose a model in which mobile code is used to obtain messages that contain its own routing code. This way we can offer applications the opportunity to differentiate themselves from one another and use the same network of different, specific, forms. From now on, there will be no need to treat different messages the same way.

Using a routing code carried by the messages to provide a blind routing where all applications dispose of the same, or no, information to make decisions would not be useful. It is necessary to let the applications use their own information in order to achieve the desired flexibility. This information should be application-oriented and designed to allow messages to decide the best route based on the specific criteria of each application.

Without the appropriate security model, this whole model is unusable. We cannot rely on a network if any malicious message could access, delete or modify all the information stored in nodes, thereby disturbing the proper operation of one or more applications. If the information used during routing becomes poisoned, then the whole architecture will loose all its purpose. Therefore, it is crucial to dispose of a system that secures the information and prevents possible damaging attacks occurring, a system that makes feasible this new approach.

The obvious solution for this problem is called access control. However, almost all access control proposals are thought to protect resources as an abstract term. This approach is general and can be used in many scenarios, with many different objectives, but the drawback of that generalization is that it restricts the tools that can be used to protect resources. For example, access control can be used to protect a file, the usage of a printer or the usage of the network, but we cannot cypher a printer or sign the usage

of the network. Besides, access control systems usually rely on Public Key Infrastructure (PKI) [2], a requirement that cannot be met in a Delay Tolerant Network.

Therefore, an access control system that would grant only messages authorized by the application access to the information owned by that application is required. This system has to take into account:

- The system has to be capable of operating in a DTN environment, considering all of its unique characteristics.
- Every application must decide exactly the messages that are authorized to access to the information it manages. All other messages' access to this information must be prohibited.
- The system has to grant access to authorized messages fast enough to not diminish their routing opportunities.

We propose Identity based Access Control for Pro-active Message's system, a secure access control system for messages with routing code focused on protecting a particular resource: structured and organized information used by messages during routing. In our model, each application is an abstract entity that can use a set of different mobile codes to route messages to its destination. These codes usually work together with a common objective. The network provides distributed application-related information used by the mobile codes in order to communicate between them, gather context-aware data or use it during routing, store intermediate results or receive instructions or parameters provided by the application.

Our proposal brings feasibility to the whole network architecture by granting confidentiality and integrity of information. In order to achieve this, we implement access control using two different hash functions and symmetric key encryption when a mobile code requests access information. The hash functions are applied to the code itself and are used to recover the cryptographic key needed to decrypt the information.

1.1. Structure of this paper

The following sections of this paper are structured this way: Section 2 explains the environment in which our research is focused, the new routing model of Pro-active message's based DTN. Section 3 provides an overview of the state of the art of access control for mobile code and related work available in the literature. Section 4 studies the best way to identify and authenticate which messages are authorized to access the information. Section 5 formally describes our proposal of identity based access control, while Section 6 addresses the main security aspects and issues related to our proposal. Section 7 shows an application based on our proposal over a DTN routing problem and evaluates its performance and feasibility. Finally, section 8 concludes this paper and provides some future lines of research.

DOI: 10.1002/sec

2. PRO-ACTIVE MESSAGES, A PARADIGM'S SHIFT

In this section, we will explain the environment in which our research has been undertaken. We will explain first Delay Tolerant Networks, and how the traditional approach deals with the challenges they present. Then we will focus on Pro-active messages as our central concept of a new Delay Tolerant Networks paradigm and we will study the utility of providing application related information to be used during routing. Finally, we will explain the three phases of operation of Pro-active message's based DTN.

2.1. Delay Tolerant Networks

Delay Tolerant Networks (DTN) [3] are networks where the low connectivity rates, the high and variable delays, and the impossibility to establish simultaneous end-to-end paths make communications very challenging. Delay Tolerant Networking is usually used in regions where the communication networks are unavailable or spotty, where the lack of a fixed infrastructure and the mobility of the nodes of the network allow them to be isolated from their neighbours during variable lapses of time.

The cornerstone of DTN is the store-carry-and-forward strategy [4], which is a way for turning a weakness as the mobility of the nodes into a strength. Using this strategy, nodes store and carry their messages until they opportunistically establish contacts with other nodes of the network, and they make use of these unpredictable contacts to forward the messages. This process is repeated until the messages eventually reach their destination. The simultaneous end-to-end paths that cannot be established are somehow substituted by a mix between physical distances travelled by nodes carrying messages and node-to-node transmissions when other nodes are contacted.

DTN routing protocols focus on the decision making when two or more nodes establish contact. What messages should be forwarded? How many copies of every message should be created? Is there a message not worth sending? What messages should be dropped last? etc... These questions are addressed by routing protocols, who try to ask these questions to maximize the performance of the network.

The traditional DTN approach consists in selecting a routing protocol and deploying it in every node of the network. This way all nodes behave the same way in terms of routing, and most important, all messages are treated equally all over the network.

2.2. Pro-active message based DTN

Our proposal revolves around the concept of changing the traditional approach to routing. Instead of using routing algorithms deployed in every node of the network, we move the routing code from the nodes to the messages, using mobile code as is suggested in [5, 6]. In this paradigm, nodes provide the necessary infrastructure to let the messages decide the way towards its destination, but the own message carries the routing algorithm that decides how to route it. The type of DTN where this research

Security Comm. Networks 2012; **00**:2–26
DOI: 10.1002/sec

Prepared using secauth.cls

is focused is called Pro-active message's DTN. The key concept of this networks is the **Pro-active message**. As depicted in Figure 1, a Pro-active message contains four fields:

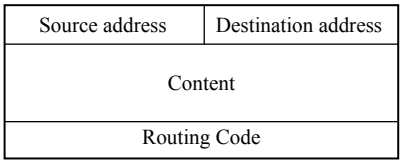

Figure 1. Schema of fields of a Pro-active message, a message that carries its own routing code.

1. **Source address**: address of the node that sent the message.
2. **Destination address**: address of the node where the message has to be delivered.
3. **Content**: the data from the application.
4. **Routing Code**: mobile code that has to be executed in every node the message arrives to in order to choose where the message should be forwarded to. Routing code executes a function f which, as defined below, operates above the list of the current neighbours and a set of contextual information and returns the subset of neighbours where the message has to be forwarded to.

$$\text{forwardTo} = f(\text{neighbourList}, \text{information})$$

$$\text{forwardTo} \subseteq \text{neighbourList}$$

2.2.1. Application driven routing

The most important advantage of Pro-active message's DTN approach is that applications themselves can use the network any way they want. It is important to note that every application knows its own needs better than any other, and similarly, the best routing decision for an application can be different to the best routing decision for another application and nobody but the applications themselves can know that. Consequently, by using Pro-active messages, applications can take their own optimal routing decisions.

Very different applications can coexist inside the same network. However, Pro-active messages cannot take optimal decisions without having enough information. In this model, the usage of an application-related information is crucial. Routing cannot be driven by applications unless they can decide everything about the information they want to use. The way to achieve this is to let the applications use ontologies to structure their information. Then, the same information is seen simultaneously in two very different ways: while the network sees it as a stream of bytes the applications see it as a structured knowledge. In our proposal of Pro-active message's based DTN, this routing information is stored in nodes in the Routing Information DataBase (RIDB). Eventually, nodes can exchange entries of the RIDB between them in order to allow applications to spread their updates of the entries.

Therefore, Pro-active messages must access their application related information in every node of their

DOI: 10.1002/sec

route. Pro-active messages must also be able to modify the information, updating its value in a way that can be known or understood by the application itself.

2.2.2. Operation of the network

During the operation of the Pro-active message's based DTN, we can differentiate between three phases, depending on the type of the messages sent and the actions performed by the users, the applications and the administrators of the network.

1. **Deployment phase:** nodes become initialized, and no Pro-active messages are sent. Network administrators must be able to access all the nodes. Ideally, deployment should be applied only once before users and applications start using the network.
2. **Standard phase:** Pro-active messages sent by users are the only kind of message that travels through the network. Nodes can become isolated during lapses of time, network topography varies quickly and there are not simultaneous end-to-end paths between nodes. This is the most common phase.
3. **Update phase:** applications update their information entries and nodes exchange these entries between them in order to spread the updates of the routing information. Applications share the network with users during this phase. Therefore, Pro-active messages continue travelling through the network. This phase starts every time an application decides to share its update and ends when all nodes have received it or after a determined amount of time.

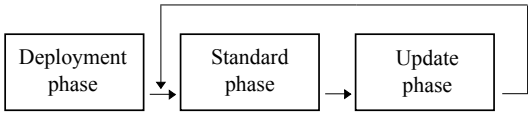

Figure 2. Flow diagram of the three phases.

Figure 2 shows the flow chart of the three phases mentioned above, the deployment phase is the initial phase of the network, followed by the standard and update phases which form an endless cycle.

3. STATE OF THE ART

In this section, we introduce the concept of mobile code related with access control. Following, we present some of the most representative research community's solutions for access control for mobile code and analyse their characteristics. We focus especially on those characteristics that can become problematic in a DTN.

3.1. Access Control for Mobile Code

Mobile code [7] is code sourced from remote, possibly unknown or untrusted systems or networks, but executed locally on the system. Some examples of mobile code are mobile agents [8], downloadable code, executable content, active capsules, remote code, etc... Due to its own nature, mobile code does not fit well with the traditional access control approach that runs any process with the same user privileges or capabilities that executed it.

DOI: 10.1002/sec

In the following paragraphs, we provide an overview of some of the ideas and proposals from the research community currently used to solve the access control for mobile code problem.

3.2. Role-Based Access Control

Role-Based policies [9], also referred to as role-based security, is the most extended and used approach in medium and large organizations [10]. Role-based policies regulate users' access to the information depending on the activities they carry out in the system. Roles are defined on the basis of the actions associated with a concrete working activity. Then, instead of giving authorizations to each user every time they want to access a resource, users are given authorization to adopt roles. Finally, each requested access is allowed or denied depending on the roles adopted by the user, Figure 3 illustrates this process.

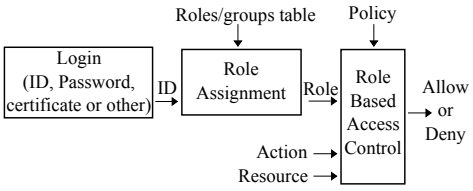

Figure 3. Schema of Role Based Access Control.

It is important to note that Role Based Access Control (RBAC) is designed to be used inside a closed system. Using it in a DTN is problematic because the access control module needs a fair role assignment. This is hard to achieve if we have to separate the access control module and the role assignment module into different nodes that could become unlinked them during the access control process. As a result, although RBAC does not use any cryptographic tool, it can not be used in a DTN schema without using cryptography to secure transmissions.

3.3. Access Control based on trust

Cryptographic tools such as digital signature [11] are used by software authors to sign the code which is distributed together with the author's digital certificate and signature. When the code arrives a host and is about to be executed, the signature is validated, and then the system grants the code access to all requested resources exclusively in the case that the author of the code is trustworthy.

There are lots of proposals designed to enhance this trust model. These proposals are based on the assumption that it is impossible for a user to know all the world's trustworthy software developers and trust their programs. Trust management [12] tries to establish trust relationships between users and developers that do not know each other. The key point is not to try to establish the authorship of the code but establish the credentials of the code instead [13]. An example of Access Control based on trust is KeyNote Trust Management System [14], where users delegate into trusted credential issuers that are expected to have direct or indirect relationships with potential requesters. Other proposals [15] have dynamically updated trust relationships as more information is collected from code execution or use recommendations from peers to calculate the scale of the trust on the unknown requester [16].

DOI: 10.1002/sec

Prepared using secauth.cls

3.4. Security-by-contract

Security-by-contract [17] goes a step further and uses digital signature to link together the code, not only with the author but also with a contract [18] that specifies the actions the code will perform during its execution. This model also relies on trusting the code author, but the idea is that only permissions that are needed are granted, or at least, permissions that the author considers necessary for the code (open and write a file, create a socket, access a database, send packets to a specific domain, etc...). Thanks to security-by-contract, users are able to know exactly not only who wrote a program, but also what actions will be executed if the contract is accepted.

Of course, security-by-contract needs to use cryptography to bind the code to a contract in order to guarantee the authenticity and the origin of the contract. Otherwise, a malicious entity could write its own contract and use it to gain access to a resource from unauthorized code.

Additionally, if an application decides to delete or revoke the permissions given to a code to access some information, it can not be done with this system. When a contract is attached to a signed code, there is no way to revoke or change it unless the certificate used to sign the contract is revoked.

3.5. History-based access control

History-based access control for mobile code, such as Deeds [19], gather and store information about the actions executed and the resources accessed by the mobile code. Subsequently, this historical information is used to link the process to several pre-defined profiles (editor, browser, terminal, etc...). Finally, those profiles are used to decide about the requests made processes (e.g. a program opens a local file is labelled as "editor", but when the same program tries to open a socket, the creation of the socket is not allowed because it is an action allowed for "browsers" but not for "editors").

This approach is extremely specific to control the execution of code downloaded from an unknown source towards Internet, or another similar environment, and it is hardly applicable to other situations. History-based Access Control does not match with the requirements of the environment. This proposal is designed to work in scenarios where the resources are very different and the only danger occurs when two or more are used together, or one after another. In Pro-active message's DTN, the only action performed by codes usually is *access information*, and a "safe" or "dangerous" behaviour does not exist (based on the information accessed by the code).

3.6. Common characteristics

It is important to note that most of the access control proposals for mobile code are thought and focused on solving two specific cases: foreign code executed inside the browser, and execution of code downloaded from an unknown source from the Internet. As a result, they usually assume some weaknesses derived from the nature of the open environment in which they are designed to operate

Security Comm. Networks 2012; **00**:2–26 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

(e.g. users cannot know all programs they will execute, the amount of different programs and software developers is enormous and keeps growing, etc...).

Another important aspect to consider is that security is, in almost most cases, cryptographically based on Public Key Infrastructure (PKI), a very extended model that fits well with Internet's characteristics but not with DTN characteristics. The access to a trusted third party, to the certificate's repository or to revocation lists cannot be available in a DTN, additionally, the distribution of the certificates among nodes remains unsolved [20, 3, 21, 22, 23]. The reason for this is that PKI is based on assumptions such as permanent point-to-point connectivity or the small delays at the link layer that cannot be applied in DTN.

4. IDENTITY OF PRO-ACTIVE MESSAGES

In this section, we will see the specific necessities of the Pro-active messages identification's process and the need of that identification in order to provide access control to a given information. Then, we will discuss different ways of message authentication, and we will study the pros and cons of our solution.

4.1. Identification of Pro-active messages

Supposing that a message arrives a node and requests access to the information of the application A. It is necessary to identify and authenticate that message in order to decide if it is authorized to access the information about this application.

Since the data field can contain any value, the quantity of different Pro-active messages that an application can create is infinite. However, during routing, the data field is meaningless, it does not play any role until the message is delivered to its destination.

We have to consider a hypothetical attacker that alters a Pro-active message modifying its routing code that tries to, using a forged routing code, access or modify A's information. This is one of the points we want to fight in our research, to develop an access control system that fits with this situation.

4.2. Authentication of Pro-active messages

In general, the authentication process can be seen from four different perspectives.

1. To find out something anything that nobody else can know, for example, a password.
2. To find out something that nobody else has, for example, a key.
3. To find out how to do something in a particular way, for example, a signature.
4. To have a unique characteristic, for example, a fingerprint or a DNA chain.

Here we will examine why the first three approaches are not valid when considering the authentication problem of the Pro-active messages in a DTN environment.

DOI: 10.1002/sec

The first two points can be analysed together. A software entity cannot differentiate between *having something* and *knowing something*. A message cannot travel with physical elements as a key or a card, but it can travel with any data, as a password or a digital key. The usage of portable passwords has an important drawback: the theft of a password compromises immediately the whole system, because from that moment, the attacker could use send a malicious message with the stolen password to access a protected information from its routing code.

The third approach makes us think directly about the usage of a well-known cryptographic technique: digital signature. However, digital signature leans in the PKI, a schema that can't be applied in DTN, as we have seen in Section 3.

We find the solution to our problem in the fourth point, above. When a routing code tries to access an entry, we can determine if it is authorized by analysing it. In that case we are using something that it is inherent to the code, something that cannot be copied or stolen because it forms part of what that routing code is. This is an idea previously pointed out in [24] to manage package distributions of software.

Using a simile with conventional identity based access control system, we can say that we analyse the DNA of the message in order to identify and authenticate it, the routing code of the message, like the DNA of a living being. This way, the message does not need to *know* or *have* or *do* anything, it will be authenticated for what it *is*.

4.3. Identify code using hash

Messages can use routing codes of different lengths. In order to use routing code to decide about access control, it is preferable to manage a fixed-size element. With just a hash function applied on the routing code, it is possible to obtain a binary sequence that identifies it and, at the same time, differentiates it from any other.

Our system uses the hash of the routing code to identify messages. This way, if a message is intercepted, the only way a hypothetical attacker could use the obtained data to access information is to create a message with the obtained routing code. Regardless, a behaviour like this would not compromise the security of the system. Note that if the routing code of the new message is exactly the same of the original message then it would not cause any malicious action over the stored information.

5. IDENTITY BASED ACCESS CONTROL

In this section, we will analyse the requisites that our access control system has to satisfy. Secondly, we will present the Authorized Hashes Set, the Entries Set and the algorithms used to add content to these sets. Thirdly, we will explain the way of using these sets and the algorithms to achieve an effective access control system. Finally, we will explain the characteristics of Identity based Access Control.

Security Comm. Networks 2012; **00**:2–26
DOI: 10.1002/sec

Prepared using secauth.cls

5.1. Notation

For the sake of clarity, we provide Table I, which contains the notation used to refer to each one of the different elements that will appear in this and the next Section, and a brief description of its meaning. From now on, we will use this notation.

Notation	Meaning
i	Identifies a message.
c_i	Routing code of message i.
c_i'	Routing code forged to replace c_i.
j	Identifies an application.
I_j	Information of application j stored in the RIDB.
I_j'	An updated version of I_j.
$h()$ and $h'()$	Two different hash functions.
$E_k()$	Symmetric key encryption function of algorithm E using key k.
$D_k()$	Symmetric key decryption function of algorithm E using key k.

Table I. Notation of all elements used from now on.

5.2. Requirements

The developed system uses the hash value of c_i from each Pro-active message in order to identify it and provide control access to the protected Information I_j, which is stored at the custodian. The following requisites need to be granted:

- The system should grant access to all authorized c_i to information I_j.
- The system should grant secrecy and integrity to all protected Information I_j. No unauthorized c_i (or other processes) should be able to access or modify it.
- The system should allow nodes to send entries between them to spread among the nodes of the network the updates made by the applications.
- The system should add a minimum impact in terms of resource's consumption and execution time. This is necessary to avoid conflicts with small connectivity windows, typical in DTN scenarios.

5.3. A system based on two sets

To decide if c_i is authorized to access I_j, our proposal is based in the usage of two sets, the set of entries and the set of stored information.

Authorized Hashes Set

The Authorized Hashes Set (AHS) is the key element of our proposal, and it is the only set that has to be deployed into nodes during the deployment phase. AHS contains a collection of triplets as explained next:

$$(j, h'(c_i), E_{h(c_i)}(k_j)) \quad (1)$$

Where:

DOI: 10.1002/sec

Prepared using secauth.cls

- j identifies the application to which the Information I_j belongs, is used to identify the triplet together with the result of applying a hash algorithm to c_i.
- k_j is the symmetric key needed to cipher and decrypt the protected information I_j, and it is cyphered with E using the result of another hash algorithm over c_i as key.

Entries Set

The Entries Set (ES) is a collection of pairs as follows:

$$(j, E_{k_j}(I_j)) \tag{2}$$

Where:

- j identifies the application to which the Information I_j and allows us to make a quick search of a stored entry identified by j without trying to decrypt every entry in the set.
- I_j is the protected information, ciphered using a symmetric encryption algorithm using a key k_j that is stored in the AHS (see following subsections 5.4 and 5.5 in order to know how the key k_j is obtained and stored).

5.4. Creation of the AHS

When new information I_j has to be stored and protected in one or more nodes, Algorithm 1 is used to add to the AHS and ES sets the needed triplets and pairs.

Algorithm 1 *Storage of protected information*

Input: I_j: Information to be protected.

j: Identifier of the information.

M: Set of messages i with access to I_j.

Output: $\emptyset$

1: Generate a random key k_j.
2: Cypher $E_{k_j}(I_j)$.
3: Ad to ES the pair $(j, E_{k_j}(I_j))$.
4: **for** $i \in M$ **do**
5: Obtain routing code c_i.
6: Calculate $h(c_i)$ and $h'(c_i)$.
7: Cypher $E_{h(c_i)}(k_j)$.
8: Add $(j, h'(c_i), E_{h(c_i)}(k_j))$ to AHS.
9: **end for**
10: **return** $\emptyset$

5.5. Using the set of entries to control access to an entry

When a message i arrives to a node and its routing code c_i requests access the Information of application j, Algorithm 2 has to be applied. This algorithm recovers the keys needed to decrypt that information only if c_i is authorized to access I_j.

5.6. Modification of a protected entry

Algorithm 3 is used when a message i, previously authorized to access I_j, modifies it and decides to overwrite I_j with the newer version I'_j.

Security Comm. Networks 2012; **00**:2–26 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

Algorithm 2 *Access Control*

Input: i: Message requesting access to the information.

j: Identifier of the information.

Output: I_j: Information requested, identified by j.

1: Obtain c_i from message i.
2: Calculate $h'(c_i)$.
3: Search in AHS a triplet that matches with (j, $h'(c_i)$, *$ciphered_key*).
4: Store in *$ciphered_key* the corresponding value.
5: Calculate $h(c_i)$.
6: Decrypt $D_{h(c_i)}(E_{h(c_i)}(\$ciphered\_key)) = k_j$.
7: Search in ES a pair that matches with (j, *$ciphered_info*).
8: Store in *$ciphered_info* the corresponding value.
9: Decrypt $D_{k_j}(E_{k_j}(\$ciphered\_info)) = I_j$.
10: **if** I_j has been successfully decrypted **then**
11: **return** I_j
12: **else**
13: **return** $\emptyset$
14: **end if**

Algorithm 3 *Modification of an entry*

Input: i: Message trying to modify the entry.

j: Identifier of the entry.

I'_j: New version of I_j.

Output: **true** or **false**.

1: Execute first 9 lines of algorithm 2.
2: **if** I_j has been successfully decrypted **then**
3: Cypher $E_{k_j}(I'_j)$.
4: Remove the pair $(j, E_{k_j}(I_j))$ from ES.
5: Add the pair $(j, E_{k_j}(I'_j))$ to ES.
6: **return true**
7: **else**
8: **return false**
9: **end if**

5.7. Optimization

Algorithm 2 and Algorithm 3 are very similar (the first 9 lines are the same). Furthermore, in any situation, Algorithm 2 (used to access to an entry) has always been executed before Algorithm 3, which is used to update the entry. Therefore, Algorithm 3, the most time-consuming of the three, can be optimized and executed quicker if certain values (obtained from intermediate calculations by Algorithm 2) are kept in the memory for a while. This way we can avoid executing repeatedly the same operations.

5.8. Characteristics of Identity based Access Control system

Identity based Access Control system uses two different hash functions, the output of one of them is used to identify the subject, and the output of the other is used to cypher and decrypt the key needed to access the information.

Our proposal uses a discretionary policy, and it is designed loosely following the guidelines of an access control list (see Figure 4). This approach consists on associating each system's object with all the authorized subjects that can access it and the actions they can perform over the object. Subsequently, the list is consulted when a

DOI: 10.1002/sec

subject requests access to an object in order to know if it is authorized or not.

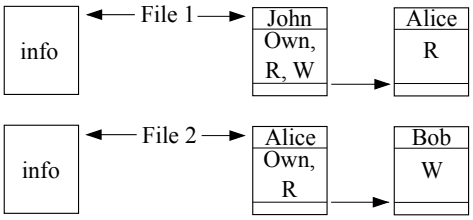

Figure 4. Schema of an Access Control Lists implementation.

Instead of relating the name of every subject with its permissions in order to allow or deny access to information, our system relates the identifier of each subject (the result of a hash function) to the key needed to decrypt the information (see Figure 5).

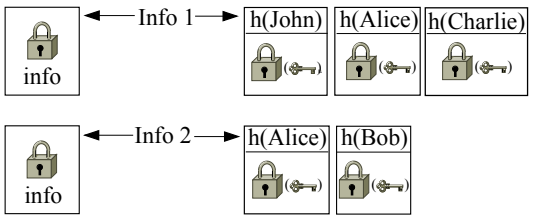

Figure 5. Schema of Identity based Access Control.

This key can be retrieved only by the proper subject (because it is encrypted with the result of another hash function is key). Moreover, the identifier of each piece of information is related with the encrypted information itself, therefore, the information can't be decrypted and accessed without getting the key.

Using the Identity based Access Control instead of simply an Access Control List system has two advantages: it provides security against a certain type of situations that we need to avoid, such as a remote attack using routing code forgery, and it allows nodes to spread updates of routing information securely (see Section 6 for more details).

5.9. Access Control in Pro-active message's DTN

We encourage the reader to look at Table II, which provides a qualitative comparison between our proposal and other access control systems. Table II is structured according to the following format: each column references one of the four compared systems (Identity based Access Control, Security-by-contract, Role-Based Access Control and History-based Access Control). Each row of the table refers to a specific characteristic of access control systems, chosen based on the needs of the environment in which we are working.

Because of this comparison, we can conclude that there are no other systems that fit well with the characteristics of Delay Tolerant Networks because they are designed to solve problems in other types of scenarios and present issues when they have to operate in a Pro-active message's DTN. We cannot find any system that improves Identity based Access Control system in that environment.

6. SECURITY OF THE ACCESS CONTROL

In this section, we will discuss security offered by our identity based access control system. With this in mind, we will analyse three different scenarios. In the first

Security Comm. Networks 2012; **00**:2–26 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

Access Control System	**Identity**	**Contract**	**RBAC**	**History-based**
Cryptographic tools used	Hash and Symmetric key encryption	Hash and Digital Signature ✗	Nothing	Nothing
Distribution of keys	Deployment	Continuous ✗	N/A	N/A
Need secure transmissions	No	No	Yes ✗	No
Application requirements	Yes	Yes	Yes	No ✗
Type of resource	Structured information	Any resource	Any resource	Any resource
State of resource	Cyphered	Original state	Original state	Original state
Initial Deployment	Platform and sets	Platform	Platform	Platform
Add new permissions	Add to set	Create new contract	Change policies	Change profiles
Delete old permissions	Remove from set	Impossible ✗	Change policies	Change profiles
Add a new resource to the system	Add to set	Add file	Change policies	Modify platform ✗
Applicability	✓	✗	✗	✗

Table II. Comparison between Identity based Access Control and other systems.

DOI: 10.1002/sec

Prepared using secauth.cls

scenario, an attacker tries to create a malicious Pro-active message that could access a protected information. In the second scenario, an attacker intercepts an exchange of routing information between nodes and tries to access these entries. In the last scenario, an attacker that has compromised a node tries to access all entries, using the sets AHS and ES. Finally, we provide some conclusions about the three scenarios.

6.1. Security against routing code forgery

A remote attacker that wants to compromise the entry I_j needs to make a Pro-active message i' with a routing code c_i' that is passed as an authorized code c_i in order to access an entry (see Figure 6).

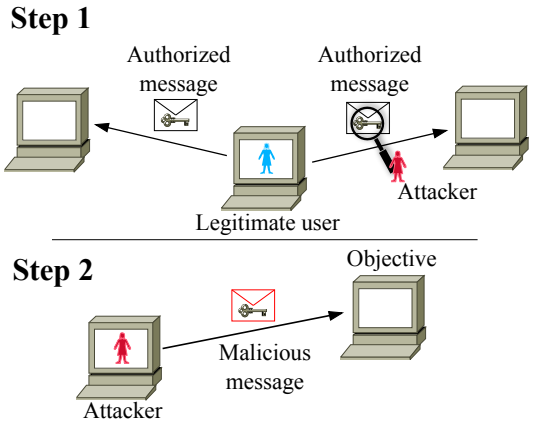

Figure 6. Schema of the attack. 1) Attacker intercepts an authorized message, 2) Attacker generates a malicious message using the information obtained in the previous step.

The first step of this attack consists of intercepting a message i with an authorized c_i that can access I_j. Then, the attacker can use the non-secret functions h and h' to calculate $h(c_i)$ and $h'(c_i)$. At this point, the attack will be successful if the attacker finds a c_i' that accomplishes both $h(c_i') = h(c_i)$ and $h'(c_i') = h'(c_i)$ with $c_i' \neq c_i$.

Therefore, the attacker has to realize a double pre-image attack [25, 26] against two different hash algorithms. Assuming that h and h' are in line with the following statements:

- They're safe against pre-image attacks. A pre-image attack consists in finding a value a such as $h(a) = b$ when b is known beforehand, and a hash function is considered safe against it if the probability of finding a value a that fits this condition is $\frac{1}{2^n}$, where n is the length in bits of the output of h (so 2^n is the total amount of possible h outputs).
- The length of the output of h and h' is n and m, respectively.
- The two algorithms are totally independent, so an attacker cannot obtain any information from either of them using the cryptanalysis of the other.

Thereby, the double pre-image attack consists in finding a c_i' such as $h(c_i') = h(c_i)$ and $h'(c_i') = h'(c_i)$. The probability of finding this value is $\frac{1}{2^n} \cdot \frac{1}{2^m} = \frac{1}{2^{n+m}}$, because both condition need to be met simultaneously.

If the attacker intercepts r different messages authorized to access the same entry, then the probability of a successful attack goes up to $r \cdot \frac{1}{2^n} \cdot \frac{1}{2^m} = \frac{r}{2^{n+m}}$ because every message authorized and intercepted provides a new target for a double pre-image attack, and all the attacks can be made in parallel to maximize the probability of success. This growth does not compromise security in any

DOI: 10.1002/sec

sense because any r possible is some orders of magnitude smaller than 2^{n+m}.

Choosing two hash functions h y h' whose output's size n and m are considered safe, and for which there are not any known algorithm that can reduce the complexity of a pre-image attack, then our system is safe against that kind of attack.

6.2. Security against update interception

In order to improve the operation of a Pro-active message's DTN, nodes spread routing information updates among the network during the update phase by sending pairs $(j, E_{k_j}(I_j))$ from the Entries Set to other nodes. This is a delicate situation, because an attacker that intercepts this transmission can obtain lots of entries of the ES.

The first thing the attacker has to do is to find the pair that contains the identifier j of the entry he wants to compromise. Next, the attack consists of breaking the ciphering provided by the symmetric key ciphering algorithm E, because there is no way to obtain the key k_j, which is stored in the AHS and has not been transferred anywhere.

If the chosen E algorithm is safe, the probability of a successful attack is $\frac{1}{2^n}$ where n is the size of the key k_j used to cipher the information (so the attack consists in trying with all possible combinations of n bits to find the key). Thereby, we can conclude that the system is safe against these kind of attacks and that nodes can send data from the ES without compromising the security of the network.

6.3. Compromising the node

This is the worst possible case. In this scenario, an attacker compromises the infrastructure of a node and wants to access all routing information of the RIDB. In this situation, the success of the attack is dependent upon acquiring Pro-active messages with authorization to access all the information. A situation like this is improbable and can compromise the security of any access control system.

The attacker tries to compromise all entries of ES using the data that can find in sets AHS and ES. The data from the AHS is not useful to the attacker unless he has intercepted some messages. When the attacker intercepts a message i with a routing code c_i that is authorized to access I_j then this entry can be compromised. In that case, the attacker can use c_i and the non-secret hash algorithms h and h' to calculate $h(c_i)$ and $h'(c_i)$ and access I_j using the algorithm explained in Section 5.

Although the system is not safe against an attack of this type, Identity based Access Control makes the success of the attack harder. Storing the information cyphered with a key that it is not present in the node forces the attacker to obtain the key using a different method, sniffing the network traffic or waiting until an authorized message arrives the node.

DOI: 10.1002/sec

6.4. Security results

We can conclude that Identity based Access Control for Pro-active Message's DTN makes the network safe against routing code forgery and update interception and message interception attacks in the active adversary mode. Thanks to Identity based Access Control, an attacker cannot create a malicious message with a routing code that compromises the security of information stored in a node by accessing it without permission.

Furthermore, nodes can easily spread updates of the routing information among the network safely, because an attacker that intercepts one or more of these updates cannot be able to decrypt it nor to access the needed keys. This contribution improves both the security of the network and the performance of its operation.

Even in the worst-case scenario attack, a compromised node where the attacker gains full control of the system, our proposal would make the success of the attack harder, and it would force the attacker to wait until the interception of some authorized messages before being able to access any cyphered information.

7. PERFORMANCE EVALUATION

To test the feasibility of our proposal, and to evaluate its operation and its performance, we have tested Identity based Access Control on a specific application. In this section, we will present the chosen application and examine the most important implementation decisions taken during the development of a proof-of-concept software of our system. Finally, we will provide some conclusions obtained from the proof-of-concept.

7.1. Scenario of application: PROSES

The year 2020 will mark a turning point in the field of European air traffic management (ATM) and control, as the next evolution in ATM is expected to become fully operational and deployed. The Single European Sky (SES) initiative will unify the heterogeneous air traffic control models used by each country, transforming the European airspace into a single integrated air management scenario. In order to achieve this, the system will require an unprecedented level of connectivity between all the participants to support the massive increase in data exchanges taking place between the terrestrial, aerial and satellite platforms. In conclusion, a sort of "aerial ATM Internet" is being created, composed of mobile and collaborative nodes that will integrate distributed and/or geographically sparse services.

The scenario where we have tested and monitored our system is based on PROSES (PROtocols for the Single European Space) [27]. In PROSES, a network of heterogeneous nodes, which in this case are aircraft and ground control centers, along with unmanned vehicles, for non-critical data exchange is created. This network is based on Pro-active message's DTN, and a dozen of different applications with varied routing needs coexist.

Security Comm. Networks 2012; **00**:2–26 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

The authors deployed a small-scale version of the proposed delay tolerant network scenario in a small aerodrome near Seville at the end of 2011 [28], where two days of flying tests were performed using two mobile nodes, located in an RC fixed-wing aircraft and an RC helicopter, and a stationary ground station. Statistics about the scenario and the characteristics of the network utilisation were collected and used here to study the feasibility of the presented Identity based Access Control system inside PROSES environment. The average number of messages carried by every node is 10, with an average size of 10KB. There are 10 different types of information with sizes between 50KB and 4MB. Field tests showed that smaller pieces of information are more commonly used than bigger pieces of information, in 82.5% of the cases access takes place with information between 100KB and 750KB. Connectivity windows in PROSES are typically located between 20 and 30 seconds.

7.2. Implementation decisions

In order to implement a Pro-active message's DTN, we have modified MobileC [29], a standard IEEE FIPA compliant mobile agent's platform to allow its use as the central element of a Pro-active message based DTN. MobileC was chosen because it is specially designed for real-time and resource constrained applications, and because it provides support to the execution of mobile code. The modifications are based in the usage of the libraries OpenSSL [30], Libxml2 [31] and Raptor [32]. The most important modifications are:

- Change of approach: MobileC is no longer a mobile agent's platform, now is an element that can send, receive and route Pro-active messages with its own routing code.
- Implementation of a module that performs the neighbour discovery. This module is very necessary to use the platform in DTN environments where the neighbour's list of any node could change very quickly.
- Inclusion of the Identity based Access Control system presented in this paper, implemented using the algorithms described in Section 5.

The chosen hash and symmetric key cypher algorithm are: $h =$ SHA2-256, $h' =$ SHA1, $E =$ AES-256 CBC. This choice was made taking into account that h and h' must be independent algorithms. Besides, the size of the key of E and the output of h was considered in order to avoid having to truncate or pad these values. Notice that they're related because $h(c_i)$ is used as a key of algorithm E when cyphering and decrypting $E_{h(c_i)}(k_j)$.

7.3. Implementation of the routing information

We have defined a collection of routing information related to different applications, which is a set of files with RDF statements. Each statement can be stored in an unprotected form, which means it is a public entry formatted as plain text, or can be protected by our system, which means that

DOI: 10.1002/sec

it has to be decrypted before it is accessed. Both public and protected entries are stored in the corresponding file using Base64. This is a way to assure that both kinds of entries are accessed the same way and the results and differences of execution's times obtained in experimentation are not affected by the parsing process.

Box 1 shows an example of a public entry and a cyphered one.

```
<rdf:Description rdf:about="http://
    test.com/publ">
  <field:data1>Value1</field:data1>
  <field:cyphered> 0 </field:cyphered>
</rdf:Description>
<rdf:Description rdf:about="http://
    test.com/priv">
  <field:data2>h+Iy3+RwDfn/qcPEF2Y5oA=
     </field:data2>
  <field:cyphered> 1 </field:cyphered>
</rdf:Description>
```

Box 1: Example of cyphered and unencrypted entries.

7.4. Deployment

Figure 7 shows the deployment diagram of the developed proof-of-concept software. As seen above, Pro-active MobileC is made up of three main elements: the neighbour discovery module, the Identity based Access Control system and the original MobileC core. The two sets AHS and ES are stored in two different files, and there is a collection of RDF [33] files that contain the routing information. These files cannot be accessed directly by Pro-active messages, and they need to send an "information request" to the platform when they want to consult the information.

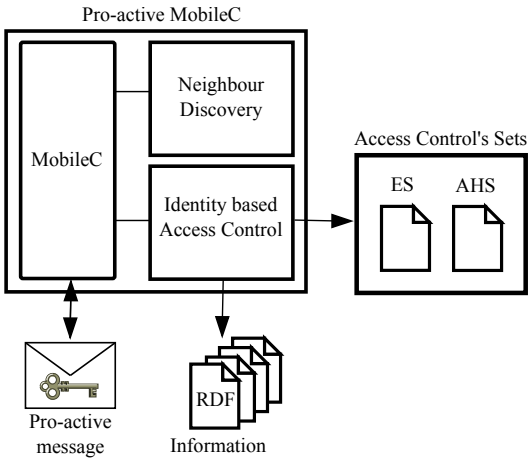

Figure 7. Deployment diagram of Pro-active MobileC.

7.5. Test environment

The laboratory environment where we have tested our system is a machine with an Intel Pentium 4 3.3GHz processor of 32 bits and 512MB of RAM memory and a GNU/Linux O.S. with a 3.3.2 kernel. The node is equipped with the Pro-active MobileC platform, and it is connected to a network interface through which it receives and sends Pro-active messages.

We have modelled the incoming traffic of Pro-active messages we injected in that node in the basis of the statistics exposed in the previous paragraph and information obtained in PROSES field tests. Figure 8 shows a schema of the operation of the test.

DOI: 10.1002/sec

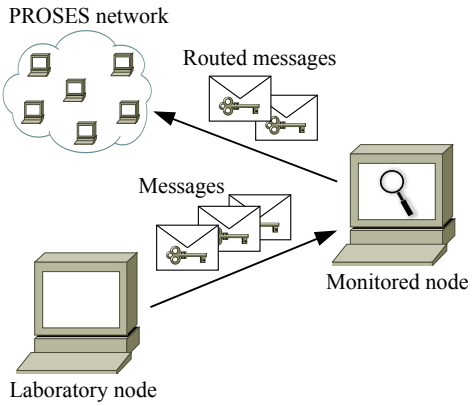

Figure 8. Schema of the proof-of-concept tests.

In order to obtain conclusive results, we have routed 2,200 messages, and we have measured the routing time spent by each one. Half of these messages were routed using public entries, while the other half was routed using protected entries, routing codes of those messages which have a structure like the one shown in Algorithm 4.

Algorithm 4 *Structure of message's routing code*

Input: destination: Message ultimate destination.

identifier: Identifier of the information.

Output: nextHop: Where to forward the message.

```
1: information = getInformation(identifier, this)
2: // Process the information to decide the next hop
3: // ...
4: return nextHop or ∅
```

Long explanations Comma splice The sizes of the entries range included: 50KB, 100KB, 200KB, 300KB, 500KB, 750KB, 1MB, 2MB, 3MB, 4MB and 5MB. Measurements allowed us to obtain average routing times and their corresponding standard deviations. We can evaluate the impact of our system by analysing the routing time taken by each message according to the size and the type of information accessed during routing.

7.6. Results

To obtain Figure 9, we have grouped the samples according to the size and the type (cyphered or unencrypted) of the accessed information. We can observe how the overhead included by our access control system becomes linear as the size of the information increases. Also, we can see that the time needed for routing when the accessed entry is cyphered is slightly longer (6.67% in average) than when it is unencrypted.

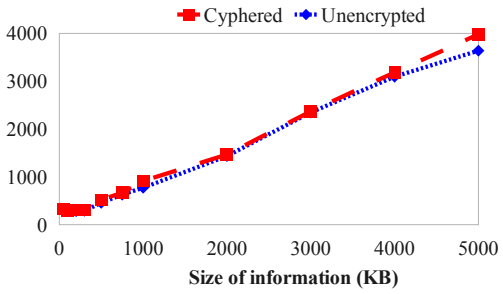

Figure 9. Routing time (ms) according to the size and the type of the accessed entry.

The obtained average routing times and its standard deviations for access to unencrypted information are shown in Table III. Table IV shows the same information for access to cyphered entries. In both cases, we have calculated the average number of milliseconds per KB taken during access by dividing the Average routing time (ms) by the Size of the entry (KB).

The amount of time spent per KB is especially important to be able to see the performance's trend of the system for

DOI: 10.1002/sec

Prepared using secauth.cls

Size (B)	Average routing time (ms)	Standard deviation	Time per KB (ms/KB)
50K	326.25	6.20%	6.53
100K	299.99	6.00%	3.00
200K	285.08	5.79%	1.43
300K	290.33	5.77%	0.97
500K	457.31	4.56%	0.91
750K	629.03	6.34%	0.84
1M	771.05	4.63%	0.75
2M	1434.15	12.72%	0.70
3M	2346.62	13.60%	0.76
4M	3092.70	8.63%	0.76
5M	3636.26	8.16%	0.71

Table III. Routing times when the accessed entry is unencrypted

Size (B)	Average routing time (ms)	Standard deviation	Time per KB (ms/KB)
50K	335.61	5.88%	6.71
100K	306.35	5.78%	3.06
200K	308.06	6.02%	1.54
300K	307.91	5.64%	1.03
500K	523.24	6.45%	1.05
750K	667.38	5.68%	0.89
1M	912.76	5.12%	0.89
2M	1465.50	9.99%	0.72
3M	2369.94	11.64%	0.77
4M	3181.93	13.75%	0.78
5M	3978.00	9.69%	0.78

Table IV. Routing times when the accessed entry is cyphered

larger sizes of information. Almost all operations executed during the access of the information are independent of its size. Thereby, we can see that when the information accessed is small, each KB is very costly due to the overhead of these operations. However, from 750KB the time per KB is stabilized around the 70-80 ms/KB (a few milliseconds more for the cyphered entries).

Figure 10 shows the routing time according to the size of the cyphered entry of the most commonly used sizes in PROSES, from 100KB to 750KB. We can see that the needed routing time is stabilized during the first half and then grows linear during the second half. Considering the size of PROSES' connectivity windows (10-30s), differences of 0.3s cannot be considered as significant, even if an increment of 0.3s represents a 120% growth in routing time (note that, in the same interval, the size of the entry has grown a 650%).

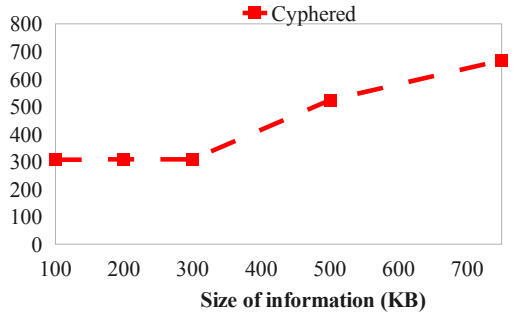

Figure 10. Routing time (ms) according to the size of the entry.

Security Comm. Networks 2012; **00**:2–26 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

The average routing time without accessing any routing information is 4.8ms (with a huge standard deviation of 39%). That time grows to 0.3-0.6s when messages access cyphered entries of 100-750KB. Considering the PROSES characteristics previously explained, we conclude that the overhead introduced by our system is totally acceptable. 10 messages that access cyphered entries of less than 1MB can be routed in less than 10 seconds, one half of the smallest connectivity window. Identity based Access Control system could be used even if the size of the used information or the number of queued messages grows, e.g. 10 messages that access entries of 4MB can be routed in 31.8s, and up to 22 messages that access entries of 500-1MB can be routed in less time than the minimum connectivity window.

Therefore, we can say our proposal is feasible, and its performance is good enough to be used in the studied DTN application, besides, as explained in Section 6, the usage of Identity based Access Control for Pro-active message's DTN would make the PROSES network safe against routing code forgery and update interception.

8. CONCLUSIONS AND FUTURE WORK

Most of DTN routing algorithms do not take into account the close relationship between applications and the way these use the network. DTN based on Pro-active messages solve that problem by allowing applications to define their own routing code and the information they need to consult during routing. This way, every application can chose differently the best way to get to its destination, and can even decide how to spread over the network creating copies of the messages. This is what we call "application driven routing".

In this paper, an access control system for routing information in a Pro-active message's based DTN has been presented. The fundamental contribution of this proposal lies in the usage of the own identity of the routing code that tries to access the information in order to decide if it is authorized. This system uses two different hash functions, one to identify the routing code that requests access to information, and the other to, together with a symmetric cyphering algorithm, protect the information and grant its confidentiality and integrity.

This proposal will not only improve security in Pro-active messages based DTN, but also make the whole paradigm become feasible. Thanks to Identity based Access Control, an attacker will not be able to create a malicious message with a routing code that compromises the security of information stored in a node by poisoning it. Furthermore, nodes can spread updates of the routing information among the nodes of the network safely. In the worst-case scenario attack, our proposal would not provide security if an attacker compromised the whole infrastructure of the attacked node. However, it would make the success of the attack harder, forcing the attacker to acquire (by intercepting authorized messages) the keys

DOI: 10.1002/sec

to access cyphered information. So it provides a last line of defence even in a situation where all other systems fall.

The application of our proposal in a specific context, PROSES, allowed us to evaluate the feasibility and the performance of the Identity based Access Control system. In conclusion, the performance of the system is good enough to be used in the Pro-active message's DTN scenario.

We also studied the characteristics of our proposal and other access control systems in order to analyse if they can be applied to the environment of our research, Pro-active message's DTN. We found that only Identity based Access Control system can be used properly in that environment due to its unique characteristics and design.

Future lines of research include, but are not limited to: widening the access control system to make it usable out of DTN scope; developing a mechanism to spread the information and enhance the operation of the network; and improving the system to allow *a posteriori* modifications of the set of rules used to decide which codes are authorized or not to access information. It is also possible to use the same principles used here to control mobile agent's access to local resources in DTN scenarios.

ACKNOWLEDGEMENT

This work has been partially funded by the Science and Innovation Ministry of Spain towards the reference project TIN2010-15764.

REFERENCES

1. C. Diot and *et al.* Haggle project. `http://www.haggleproject.org/`.
2. Amir Herzberg, Yosi Mass, Joris Michaeli, Yiftach Ravid, and Dalit Naor. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, pages 2–, Washington, DC, USA, 2000. IEEE Computer Society.
3. S. Farrell and V. Cahill. *Delay- and Disruption-Tolerant Networking*. Artech House, Inc., Norwood, MA, USA, 2006.
4. K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), November 2007.
5. S. Castillo, S. Robles, M. de Toro, and J. Borrell. Seguridad en protocolos de encaminamiento para redes dtn. In *Actas de la XI Reunión Española de Criptología y Seguridad de la Información*, Tarragona, September 2010.
6. C. Borrego and S. Robles. Seguridad en la planificaciónn de agentes móbiles en redes dtn. In *Actas de la XI Reunión Española de Criptología y Seguridad de la Información*, Tarragona, September 2010.
7. Gianpaolo Cugola, Carlo Ghezzi, Gian Picco, and Giovanni Vigna. Analyzing mobile code languages. In Jan Vitek and Christian Tschudin, editors, *Mobile*

DOI: 10.1002/sec

Prepared using secauth.cls

Object Systems Towards the Programmable Internet, volume 1222 of *Lecture Notes in Computer Science*, pages 91–109. Springer Berlin / Heidelberg, 1997. 10.1007/3-540-62852-5_9.

8. D. B. Lange and M. Oshima. Seven Good Reasons for Mobile Agents. *COMMUNICATIONS OF THE ACM*, 42(3), March 1999.
9. Ravi S. Sandhu. Role-based access control. volume 46 of *Advances in Computers*, pages 237 – 286. Elsevier, 1998.
10. A. C. O'Connor and R. J. Loomis. Economic Analysis of Role-Based Access Control . Technical report, National Institute of Standards and Technology, December 2010.
11. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.
12. Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *PROCCEDINGS OF ITRUST 2005, NUMBER 3477 IN LNCS*, pages 77–92. Springer-Verlag, 2005.
13. Sudip Chakraborty and Indrajit Ray. Trustbac: integrating trust relationships into the rbac model for access control in open systems. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, SACMAT '06, pages 49–58, New York, NY, USA, 2006. ACM.
14. Matt Blaze, John Ioannidis, and Angelos D. Keromytis. Experience with the keynote trust management system: Applications and future directions. In *In Proceedings of the 1st International Conference on Trust Management*, pages 284–300. Springer-Verlag, 2003.
15. Ching Lin, Vijay Varadharajan, Yan Wang, and Vineet Pruthi. Trust enhanced security for mobile agents. In *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on*, pages 231 – 238, july 2005.
16. Pho Duc Giang, Le Xuan Hung, Sungyoung Lee, Young-Koo Lee, and Heejo Lee. A flexible trust-based access control mechanism for security and privacy enhancement in ubiquitous systems. *Multimedia and Ubiquitous Engineering, International Conference on*, 0:698–703, 2007.
17. N. Bielova, N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan. Matching in security-by-contract for mobile code. *Journal of Logic and Algebraic Programming*, 78(5):340 – 358, 2009. The 1st Workshop on Formal Languages and Analysis of Contract-Oriented Software (FLACOS'07).
18. Richard Helm, Ian M. Holland, and Dipayan Gangopadhyay. Contracts: specifying behavioral compositions in object-oriented systems. *SIGPLAN Not.*, 25(10):169–180, September 1990.
19. Guy Edjlali, Anurag Acharya, and Vipin Chaudhary. History-based access control for mobile code. In

Secure Internet Programming, pages 413–431, 1999.

20. N. Asokan, K. Kostianinen, P. Ginzboorg, J. Ott, and C. Luo. Towards Securing Disruption-Tolerant Networking. *Nokia Research Center, Tech. Rep. NRC-TR-2007-007*, 2007.
21. K. Aniket, Z. Gregory M., and H. Urs. Anonymity and Security in Delay Tolerant Networks. In *Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, pages 504–513, 2007.
22. A. Seth and S. Keshav. Practical Security for Disconnected Nodes. In *1st IEEE ICNP Workshop on Secure Network Protocols, 2005.(NPSec)*, pages 31–36, 2005.
23. N. Asokan, K. Kostianinen, P. Ginzboorg, J. Ott, and C. Luo. Applicability of Identity-Based Cryptography for Disruption-Tolerant Networking. In *MobiOpp '07: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pages 52–56, New York, NY, USA, 2007. ACM.
24. Jeffrey K. Hollingsworth and Ethan L. Miller. Using Content-Derived Names for Caching and Software Distribution. Technical report, Technical Reports of the Computer Science Department, October 1998.
25. P. Hoffman and B. Schneier. Attacks on Cryptographic Hashes in Internet Protocols. RFC 4270, November 2005.
26. J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2^n work. Cryptology ePrint Archive, Report 2004/304, 2004. http://eprint.iacr.org/.
27. N. Giuditta, S. Robles, A. Viguria, S. Castillo, M. Cordero, and L. Fernández. Proses - network communications for the future european atm system. In *Proceedings of the International Conference on Application and Theory of Automation in Command and Control Systems*, May 2011.
28. R. Martínez-Vidal, S. Castillo-Pérez, S. Robles, A. Sánchez-Carmona, J. Borrel, M. Cordero, A. Viguria, and N. Giuditta. Mobile-agent based delay-tolerant network architecture for non-critical aeronautical data communications. In *Distributed Computing and Artificial Intelligence*, volume 217 of *Advances in Intelligent Systems and Computing*, pages 513–520. Springer International Publishing, 2013.
29. MobileC. `http://www.mobilec.org/`.
30. OpenSSL Project. `http://www.openssl.org/`.
31. The XML C parser and toolkit of Gnome. `http://xmlsoft.org/`.
32. Raptor RDF Syntax Library. `http://librdf.org/raptor/`.
33. G. Klyne and J.J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, February 2004.

DOI: 10.1002/sec

Prepared using secauth.cls